



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Trefor Southwell et al.
Application No. : 10/072,814
Filed : February 8, 2002
For : EVALUATION AND OPTIMISATION OF CODE

Art Unit : 2185
Docket No. : 858063.456
Date : May 13, 2002

Box Missing Parts
Commissioner for Patents
Washington, DC 20231

PRELIMINARY AMENDMENT

Commissioner for Patents:

Please amend the above-identified patent application as follows:

In the Headings:

- Please reformat main PTO headings to remove underlining and to change case from first letter capitalization to all uppercase.

In the Title:

- Please reformat and amend the title of the application to read as follows.

EVALUATION AND OPTIMIZATION OF CODE

In the Abstract:

- Please reformat the heading on line 1 of the Abstract page (page 14) as follows:

ABSTRACT OF THE DISCLOSURE

- Please delete the title on line 2 of the Abstract page (page 14).
- Please amend the Abstract to read as follows.

A memory map evaluation tool is provided that organizes a program in a manner most compatible with use of a cache. The tool includes a method that involves executing a first version of the program according to a first memory map to generate a program counter trace, converting the program counter trace into a specific format and then translating the program counter trace into physical addresses using a memory map to be evaluated, different from the first memory map. Those physical addresses are then used to evaluate the number of likely cache misses using a model of a direct-mapped cache for the memory map under evaluation.

In the Specification:

- Please delete the heading on page 1 at line 1.
- Please add the following new heading on page 1, between lines 2 and 3:

BACKGROUND OF THE INVENTION

- Please amend the paragraph beginning on page 1 at line 4 to read as follows:

The present invention relates to the evaluation and optimization of code, particularly to be used in a processor including a cache.

- Please amend the heading on page 1 at line 6 to read as follows:

Description of the Related Art

- Please amend the paragraph beginning on page 1 at line 10 to read as follows:

Caches are high-cost, high-speed memories that provide an important performance optimization in processors. This is done by keeping copies of the contents of most commonly used locations of main memory near to the processor, namely in cache locations. As a result, accesses to the contents of these memory locations are much quicker.

- Please amend the paragraph beginning on page 1 at line 15 to read as follows:

The instruction cache is responsible for optimizing accesses to the program being executed. The cache will usually be smaller than the size of the program, meaning that the contents of the cache will need to change to ensure that the parts of the program currently being executed are in the cache.

- Please delete the paragraph beginning on page 3 at line 18.

- Please reformat and amend the heading on page 3 at line 21 to read as follows:

BRIEF SUMMARY OF THE INVENTION

- Please amend the paragraph beginning on page 3 at line 22 to read as follows:

According to the disclosed embodiments of the invention, there is provided a method of evaluating a set of memory maps for a program comprising a plurality of functions, the method comprising: (a) executing a first version of the program according to a first memory map to generate a program counter trace; (b) converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map; (c) translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map; (d) evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map; and repeating steps (c) and (d) for each of the memory maps in the set.

- Please reformat and amend the heading on page 5 at line 4 to read as follows:

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

- Please reformat and amend the heading on page 5 at line 13 to read as follows:

DETAILED DESCRIPTION OF THE INVENTION

- Please amend the paragraph beginning on page 6 at line 1 to read as follows:

The potential difficulty with the direct-mapped cache 4 that does not exist in the four way set associative cache can readily be seen from Figure 1. That is, if block 1 is in the

direct-mapped cache 4 (at line 1) and then block 513 is to be executed, the only location in the cache suitable for accepting block 513 is line 1, which requires the eviction of block 1. If block 1 (or indeed block 513) is not often used, this is probably not too much of a problem. However, in programs where block 513 is often used, and in particular is often used after block 1, this requires more or less constant cache eviction and replacement which affects performance and increases bus traffic as discussed above.

- Please amend the paragraph beginning on page 6 at line 10 to read as follows:

Figure 2 is an example of an MPEG decoder application stored in a main memory 2 and including a variable length decode function (VLD) and an inverse discrete cosine transform (IDCT). Assume, as shown by the arrows, that these functions relate to blocks which map onto the same line or lines in the instruction cache 4. Due to the frequent usage of these functions within the decoder application, this would be a situation where a direct-mapped cache would be ineffective.

- Please amend the paragraph beginning on page 6 at line 18 to read as follows:

In brief, the tool changes the memory map of a program in order to minimize conflicts and hence increase performance. Creating a new memory map simply means placing the functions in a new order in memory. legend

- Please amend the paragraph beginning on page 7 at line 4 to read as follows:

An extremely effective method of optimizing the mapping for the instruction cache relies on the ability to generate traces of the Program Counter (PC) as the program 3 executes on a typical data set 5. Figure 4 illustrates a memory mapping tool 6 which works in this way where the execution is denoted by an execute block 7, and Figure 5 in a flow diagram.

- Please amend the paragraph beginning on page 8 at line 1 to read as follows:

The tool 6 uses this trace format to explore new memory maps (labelled Memory Map 1 (10), Memory Map 2 (10'), and memory map 3 (10'') in Figure 4), looking for one that generates the minimum number of instruction cache misses. This process of exploration has the

advantage that the time to evaluate each memory map is much quicker than actually re-linking and benchmarking the program.

- Please amend the paragraph beginning on page 8 at line 23 to read as follows:

At the start, each of the memory maps in the set is randomized. Then the tool iterates until the end criteria are met.

- Please amend the paragraph beginning on page 10 at line 3 to read as follows:

This software optimization method is not guaranteed to work for all applications, but there are many suitable applications where this optimization method can be used effectively, allowing direct-mapped caches to be used.

- Please amend the paragraph beginning on page 10 at line 6 to read as follows:

Essentially, optimizing a program for the instruction cache will work well if the program demonstrates repeatable execution flow. This is true of many streaming data (audio/video) applications, where typical data sets can be used to determine the execution flow of the application.

- Please add the following new paragraph on page 10, following line 9:

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.

In the Claims:

- Please add the following new heading on page 11 at line 1:

CLAIMS

- Please amend claims 1-6 and 8 to read as follows:

1. (Amended) A method of evaluating a set of memory maps for a program having a plurality of functions, the method comprising:

(a) executing a first version of the program according to a first memory map to generate a program counter trace;

(b) converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map;

(c) translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map;

(d) evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map; and

repeating steps (c) and (d) for each of the memory maps in the set.

2. (Amended) The method of claim 1, wherein step (c) is carried out by utilizing the base address of each function of said one of the memory maps to be evaluated with the offset given in the program count trace format.

3. (Amended) The method of claim 1, wherein the direct-mapped cache model of step (d) emulates the operation of a cache such that would occur when a new version of the program linked according to said one memory map under evaluation is executed.

4. (Amended) The method of claim 1, comprising the additional step of, subsequent to evaluating the first set of memory maps, generating a further set of memory maps for evaluation.

5. (Amended) A method of operating a computer to evaluate a set of memory maps for a program comprising a plurality of functions, the method comprising:

loading a first version of the program into the computer and executing said first version to generate a program counter trace;

loading into the computer a memory map evaluation tool that carries out the steps of:

converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map;

translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map; and

evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map;

wherein the step of translating a program counter trace and evaluating the number of likely cache misses is repeated for each of the memory maps in a set to be evaluated.

6. (Amended) The method of claim 5, wherein the memory map generation tool is also operable to generate a further set of memory maps for evaluation taking into account the results of evaluation of the first set of memory maps.

8. (Amended) The tool of claim 7, configured in the form of program code means which, when executed on a computer, carry out the method steps of:

(a) executing a first version of the program according to a first memory map to generate a program counter trace;

(b) converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map;

(c) translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map;

(d) evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map; and

repeating steps (c) and (d) for each of the memory maps in the set.

- Please add new claims 9-17 to read as follows:

9. (New) A method of optimizing memory mapping for an instruction cache, comprising:

generating a plurality of alternate memory maps for evaluation;

evaluating each of the plurality of alternate memory maps for the potential number of missed instructions resulting from cache memory conflicts; and

selecting at least one of the evaluated plurality of alternate memory maps having the fewest potential number of missed instructions.

10. (New) A method of optimizing memory mapping for an instruction cache having an original memory map, comprising:

generating a plurality of alternate memory maps;

evaluating each of the plurality of alternate memory maps for the potential number of missed instructions resulting from cache memory conflicts;

selecting at least one of the evaluated plurality of alternate memory maps and in accordance with predetermined criteria;

generating new alternate memory maps from the selected at least one of the evaluated plurality of alternate memory maps and the original map;

evaluating the new alternate memory maps; and

repeatedly generating and evaluating new alternate memory maps until an end criteria is met.

11. (New) The method of claim 10, wherein the end criteria comprises one from among: a predetermined number of memory maps that have been evaluated, failure to find a better memory map, and when a member of missed instructions is greater than a predetermined minimum number of missed instructions.

12. (New) The method of claim 10, wherein generating new alternate memory maps comprises one from among the following: swapping functions from the selected at least one of the plurality of alternate memory maps and the original memory map, and selecting at least two of the alternate memory maps and merging the selected at least two of the alternate memory maps with the original memory map.

13. (New) A software optimization method, comprising:

compiling a program;

generating a first memory map;

executing the program and generating a program count trace;

converting the program count trace to a format for finding a memory location in association with a function and an offset within the function using the first memory map; and evaluating and selecting a memory map, further comprising:

- translating the program counter trace into physical addresses;
- executing the translated trace on a cache model;
- determine the number of cache misses; and

when the number of cache misses is acceptable, selecting the memory map and relinking to the program, otherwise selecting a new memory map and returning to the substep of translating the program count trace using the new memory map.

14. (New) A software optimization method, comprising:

- selecting a plurality of memory maps for evaluation and the criteria for terminating the evaluation;

- randomizing each of the selected memory maps;

- evaluating each of the memory maps until the criteria for terminating the evaluation is met, further comprising:

- evaluating the performance of each selected memory map; and

- creating a new set of memory maps from the memory maps previously evaluated for a next evaluation.

15. (New) The method of claim 14, wherein creating a new set of memory maps comprises selecting the memory map with the least number of missed instructions resulting from cache memory conflicts and performing a swap of two random functions therein.

16. (New) The method of claim 14, wherein creating a new set of memory maps comprises selecting two or more memory maps having the best performance from the previous evaluation and merging the changes from the two selected memory maps to create a new memory map.

17. (New) The method of claim 14, wherein the end criteria comprises one from among: a set number of evaluations that have been performed, a set number of misses that

have been reached, and failure to find a better memory map after a predetermined number of evaluations.

REMARKS

In the event the Examiner finds informalities that can be resolved by telephone conference or the Examiner is unwilling to enter this amendment, applicants respectfully request that the Examiner contact the applicants' undersigned representative by telephone at (206) 622-4900 in order to expeditiously resolve prosecution of this application. Consequently, early and favorable action allowing these claims and passing this case to issuance is respectfully solicited.


Attached hereto is a marked-up version of the changes made to the specification and claims by the current amendment. The attached page is captioned "Version With Markings to Show Changes Made."

All of the claims remaining in the application are now clearly allowable. Favorable consideration and a Notice of Allowance are earnestly solicited.

Respectfully submitted,

Trefor Southwell et al.

Seed Intellectual Property Law Group PLLC



E. Russell Tarleton
Registration No. 31,800

ERT:aep

Enclosure:

Postcard

701 Fifth Avenue, Suite 6300
Seattle, Washington 98104-7092
Phone: (206) 622-4900
Fax: (206) 682-6031

261655_1 DOC

VERSION WITH MARKINGS TO SHOW CHANGES MADE**NOTE re marking conventions:**

- Reformatted text is underlined.
- Inserted text is double underlined.
- Deleted text is marked with ~~strikethrough~~.

In the Headings:

Main PTO headings have been reformatted to remove underlining and to change case from first letter capitalization to all uppercase.

In the Title:

- The title of the application has been reformatted and amended to read as follows.

EVALUATION AND ~~OPTIMISATION~~OPTIMIZATION OF CODE

In the Abstract:

- The heading on line 1 of the Abstract page (page 14) has been reformatted as follows:

ABSTRACT OF THE DISCLOSURE

- The title on line 2 of the Abstract page (page 14) has been deleted.

~~EVALUATION AND OPTIMISATION OF CODE~~

- The Abstract has been amended as follows.

A memory map evaluation tool is provided ~~which allows that~~ organizes a program to be organised in a manner most compatible with use of a cache. ~~This is done by~~ The tool includes a method that involves executing a first version of the program according to a first memory map to generate a program counter trace, converting the program counter trace into a specific format and then translating the program counter trace into physical addresses using a memory map to be evaluated, different from the first memory map. Those physical addresses are then used to evaluate the number of likely cache misses using a model of a direct-mapped cache for the memory map under evaluation.

In the Specification:

- The heading on page 1 at line 1 has been deleted:

Title of the Invention

- The following new heading has been added on page 1, between lines 2 and 3.

BACKGROUND OF THE INVENTION

- The paragraph beginning on page 1 at line 4 has been amended as follows:

The present invention relates to the evaluation and ~~optimisation~~optimization of code, particularly to be used in a processor including a cache.

- The heading on page 1 at line 6 has been amended as follows:

Background-Description of the Related Art-Invention

- The paragraph beginning on page 1 at line 10 has been amended as follows:

Caches are high-cost, high-speed memories that provide an important performance ~~optimisation~~optimization in processors. This is done by keeping copies of the contents of most commonly used locations of main memory near to the processor, namely in cache locations. As a result, accesses to the contents of these memory locations are much quicker.

- The paragraph beginning on page 1 at line 15 has been amended as follows:

The instruction cache is responsible for ~~optimising~~optimizing accesses to the program being executed. The cache will usually be smaller than the size of the program, meaning that the contents of the cache will need to change to ensure that the parts of the program currently being executed are in the cache.

- The paragraph beginning on page 3 at line 18 has been deleted:

~~It is an aim of the present invention to reduce or eliminate conflicts in a direct-mapped cache to allow advantage to be taken of the smaller area and higher clock frequencies characteristic of such caches.~~

- The heading on page 3 at line 21 has been reformatted and amended as follows:

BRIEF SUMMARY OF THE INVENTION

- The paragraph beginning on page 3 at line 22 has been amended as follows:

According to ~~one aspect~~ the disclosed embodiments of the invention, there is provided a method of evaluating a set of memory maps for a program comprising a plurality of functions, the method comprising: (a) executing a first version of the program according to a first memory map to generate a program counter trace; (b) converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map; (c) translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map; (d) evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map; and repeating steps (c) and (d) for each of the memory maps in the set.

- The heading on page 5 at line 4 has been reformatted and amended as follows:

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

- The heading on page 5 at line 13 has been reformatted and amended as follows:

DETAILED DESCRIPTION OF THE INVENTION Preferred Embodiment

- The paragraph beginning on page 6 at line 1 has been amended as follows:

The potential difficulty with ~~a~~ the direct-mapped cache ~~which 4 that~~ does not exist in ~~a~~ the four way set associative cache can readily be seen from Figure 1. That is, if block 1 is in the direct-mapped cache 4 (at line 1) and then block 513 is to be executed, the only location in the cache suitable for accepting block 513 is line 1, which requires the eviction of block 1. If

block 1 (or indeed block 513) is not often used, this is probably not too much of a problem. However, in programs where block 513 is often used, and in particular is often used after block 1, this requires more or less constant cache eviction and replacement which affects performance and increases bus traffic as discussed above.

- The paragraph beginning on page 6 at line 10 has been amended as follows:

Figure 2 is an example of an MPEG decoder application stored in a main memory 2 and including a variable length decode function (VLD) and an inverse discrete cosine transform (IDCT). Assume, as shown by the arrows, that these functions relate to blocks which map onto the same line or lines in the instruction cache 4. Due to the frequent usage of these functions within the decoder application, this would be a situation where a direct-mapped cache would be ineffective.

- The paragraph beginning on page 6 at line 18 has been amended as follows:

In brief, the tool changes the memory map of a program in order to ~~minimise~~minimize conflicts and hence increase performance. Creating a new memory map simply means placing the functions in a new order in memory. legend

- The paragraph beginning on page 7 at line 4 has been amended as follows:

An extremely effective method of ~~optimising~~optimizing the mapping for the instruction cache relies on the ability to generate traces of the Program Counter (PC) as the program 3 executes on a typical data set 5. Figure 4 illustrates a memory mapping tool 6 which works in this way where the execution is denoted by an execute block 7, and Figure 5 in a flow diagram.

- The paragraph beginning on page 8 at line 1 has been amended as follows:

The tool 6 uses this trace format to explore new memory maps (labelled Memory Map 1 (10), Memory Map 2 (10'),~~etc.~~ and memory map 3 (10'')) in Figure 4), looking for one that generates the minimum number of instruction cache misses. This process of exploration has the

advantage that the time to evaluate each memory map is much quicker than actually re-linking and benchmarking the program.

- The paragraph beginning on page 8 at line 23 has been amended as follows:

At the start, each of the memory maps in the set is ~~randomised~~randomized. Then the tool iterates until the end criteria are met.

- The paragraph beginning on page 10 at line 3 has been amended as follows:

This software ~~optimisation~~optimization method is not guaranteed to work for all applications, but there are many suitable applications where this ~~optimisation~~optimization method can be used effectively, allowing direct-mapped caches to be used.

- The paragraph beginning on page 10 at line 6 has been amended as follows:

Essentially, ~~optimising~~optimizing a program for the instruction cache will work well if the program demonstrates repeatable execution flow. This is true of many streaming data (audio/video) applications, where typical data sets can be used to determine the execution flow of the application.

- The following new paragraph has been added on page 10, following line 9:

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.

In the Claims:

- The following new heading has been added on page 11 at line 1 as follows:

CLAIMS

- Claims 1-6 and 8 have been amended as follows:

1. (Amended) A method of evaluating a set of memory maps for a program ~~comprising~~having a plurality of functions, the method comprising:

(a) executing a first version of the program according to a first memory map to generate a program counter trace;

(b) converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map;

(c) translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map;

(d) evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map; and

repeating steps (c) and (d) for each of the memory maps in the set.

2. (Amended) ~~A The method according to of~~ claim 1, wherein step (c) is carried out by ~~utilising~~utilizing the base address of each function of said one of the memory maps to be evaluated with the offset given in the program count trace format.

3. (Amended) ~~A The method according to of~~ claim 1 ~~or 2~~, wherein the direct-mapped cache model of step (d) emulates the operation of a cache ~~which such that~~ would occur ~~if when~~ a new version of the program linked according to said one memory map under evaluation were to be ~~is~~ executed.

4. (Amended) ~~A The method according to any preceding of~~ claim 1, which ~~requires comprising~~ the additional step of, subsequent to evaluating the first set of memory maps, generating a further set of memory maps for evaluation.

5. (Amended) A method of operating a computer to evaluate a set of memory maps for a program comprising a plurality of functions, the method comprising:

loading a first version of the program into the computer and executing said first version to generate a program counter trace;

loading into the computer a memory map evaluation tool ~~which that~~ carries out the steps of:

converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map;

translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map; and

evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map;

wherein the step of translating a program counter trace and evaluating the number of likely cache misses is repeated for each of the memory maps in a set to be evaluated.

6. (Amended) ~~A~~The method according to ~~of~~ claim 5, wherein the memory map generation tool is also operable to generate a further set of memory maps for evaluation taking into account the results of evaluation of the first set of memory maps.

8. (Amended) ~~A~~The tool according to ~~of~~ claim 7, configured in the form of program code means which, when executed on a computer, carry out the method steps of: ~~claim 1.~~

(a) executing a first version of the program according to a first memory map to generate a program counter trace;

(b) converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map;

(c) translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map;

(d) evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map; and

repeating steps (c) and (d) for each of the memory maps in the set.